An Interior-Point Algorithm with Inexact Step Computations

Frank E. Curtis

INFORMS Annual Meeting

October 13, 2009

Outline

Outline

Large-scale constrained optimization

- Consider large-scale problems of the form

$$\begin{aligned} \min\ & f(x) \\ \text{s.t.}\ & c^{\mathcal{E}}(x) = 0 \\ & c^{\mathcal{I}}(x) \geq 0 \end{aligned}$$

Large-scale constrained optimization

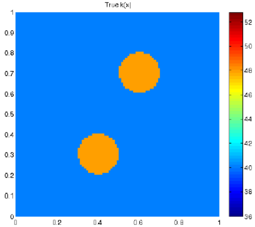
- Consider large-scale problems of the form

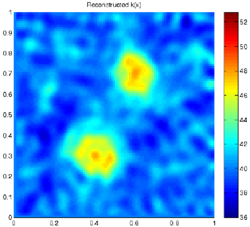
$$\begin{aligned} &\min f(x) \\ &\text{s.t. } c^{\mathcal{E}}(x) = 0 \qquad \text{(e.g., a PDE)} \\ &\qquad c^{\mathcal{I}}(x) \geq 0 \end{aligned}$$

- Problem is infinite-dimensional

Inverse problems

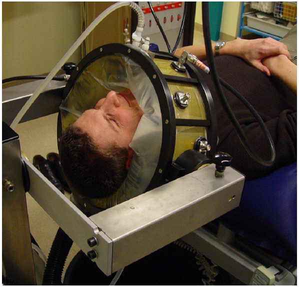
Recover a parameter k based on data collected from propagating waves





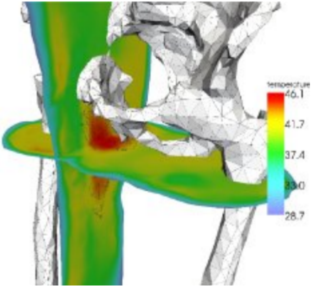
Optimal design

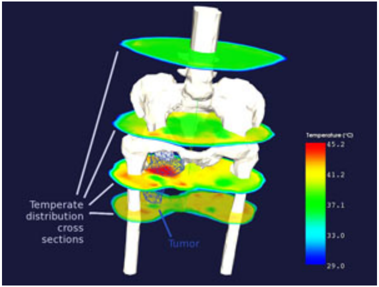
- Regional hyperthermia is a cancer therapy that aims at heating large and deeply seated tumors by means of radio wave adsorption
- Results in the killing of tumor cells and makes them more susceptible to other accompanying therapies; e.g., chemotherapy



Optimal design

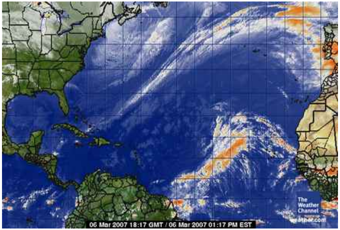
- Computer modeling can be used to help plan the therapy for each patient, and it opens the door for numerical optimization
- The goal is to heat the tumor to a target temperature of 43°C while minimizing damage to nearby cells





Data assimilation

- Weather forecasting

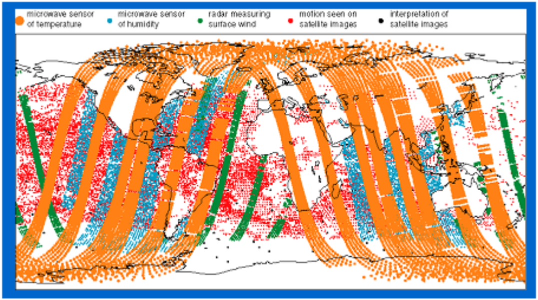


- If the initial state of the atmosphere (temperatures, pressures, wind patterns, humidities) were known at a certain point in time, then an accurate forecast could be obtained by integrating atmospheric model equations forward in time
- Flow described by Navier-Stokes and further sophistications of atmospheric physics and dynamics

Data assimilation

Limited amount of data (satellites, buoys, planes, ground-based sensors)

- Each observation is subject to error
- Nonuniformly distributed around the globe (satellite paths, densely-populated areas)



Outline

Problem reformulation

- The logarithmic-barrier subproblem:

$$\begin{aligned} \min\ & f(x) - \mu \sum_{i=1}^{q} \ln s^i \\ \text{s.t. } & c^{\mathcal{E}}(x) = 0 \\ & c^{\mathcal{I}}(x) = s \end{aligned}$$

- If f, $c^{\mathcal{E}}$, and $c^{\mathcal{I}}$ are smooth, the optimality conditions are:

$$\begin{aligned} \nabla f(x) + \nabla c^{\mathcal{E}}(x)\lambda^{\mathcal{E}} + \nabla c^{\mathcal{I}}(x)\lambda^{\mathcal{I}} &= 0 \\ -\mu S^{-1}e - \lambda^{\mathcal{I}} &= 0 \\ c^{\mathcal{E}}(x) &= 0 \\ c^{\mathcal{I}}(x) - s &= 0 \end{aligned}$$

along with $s > 0$

Newton's method

- Applying Newton's method yields the linear system

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

Usual questions

- How do we ensure global convergence?
- How do we solve ill-conditioned problems?
- How do we handle nonconvexity?

Usual answers

- How do we ensure global convergence?
  - KKT conditions (convex case)
  - Merit/penalty function
  - Filter
- How do we solve ill-conditioned problems?
  - Matrix modifications
  - Trust regions
- How do we handle nonconvexity?
  - Matrix modifications
  - Trust regions

More questions

For large-scale problems:

- What if the derivative matrices cannot be stored?
- What if the derivative matrices cannot be factored?

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

We can use iterative in place of direct methods:

- Can we incorporate inexactness?
- How do we ensure global convergence, handle ill-conditioning, and handle nonconvexity if solutions are inexact?

Outline

References

- "An Inexact SQP Method for Equality Constrained Optimization," R. H. Byrd, F. E. Curtis, and J. Nocedal, SIAM Journal on Optimization, Volume 19, Issue 1, pg. 351–369, 2008.
- "An Inexact Newton Method for Nonconvex Equality Constrained Optimization," R. H. Byrd, F. E. Curtis, and J. Nocedal, to appear in Mathematical Programming Series A.
- "A Matrix-free Algorithm for Equality Constrained Optimization Problems with Rank-Deficient Jacobians," F. E. Curtis, J. Nocedal, and A. Wächter, SIAM Journal on Optimization, Volume 20, Issue 3, pg. 1224–1249.
- "An Interior-Point Algorithm for Large-Scale Nonlinear Optimization with Inexact Step Computations," F. E. Curtis, O. Schenk, and A. Wächter, submitted to SIAM Journal on Scientific Computing.

Rank deficiency

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

If the constraint Jacobian is singular or ill-conditioned

- The system may be inconsistent
- The search directions $(d_k^x, d_k^s, \delta_k^{\mathcal{E}}, \delta_k^{\mathcal{I}})$ may blow up
- The line search may break down

A typical remedy: Matrix modification

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & -\xi I & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & -\xi I \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

A typical remedy: Matrix modification

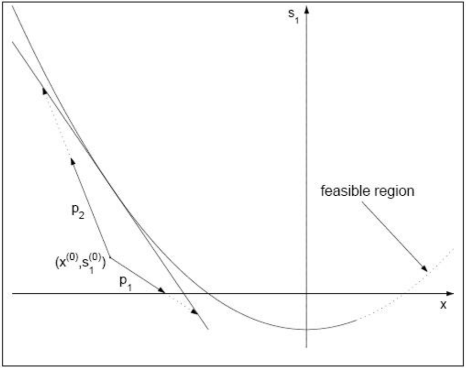
$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & -\xi I & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & -\xi I \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

However, without matrix factorizations (i.e., no idea of the inertia)

- When should this modification be performed?
- What value should ξ take? How large?
- How do we ensure that in the end we solve the right problem?

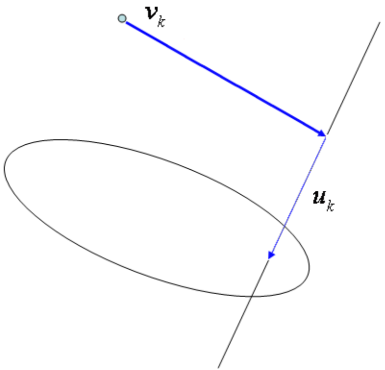
Failure of line search methods

- Recall the counter example of Wächter and Biegler (2000)

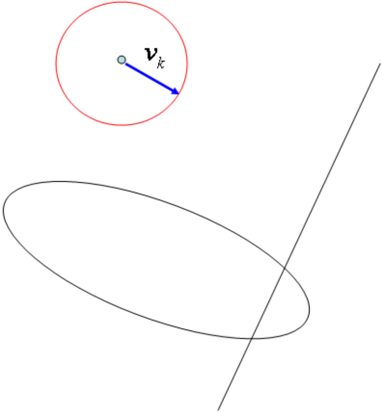


- (Graph courtesy of Nocedal and Wright, 2006)

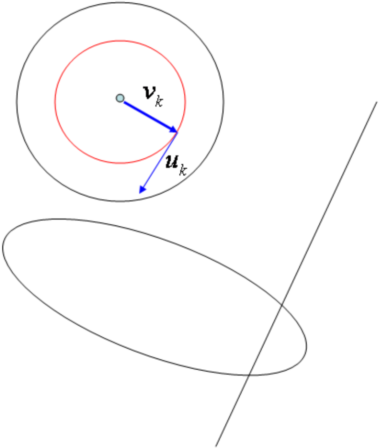
Step decomposition



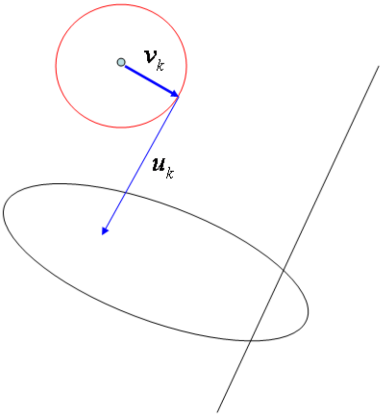
Step decomposition



Step decomposition



Step decomposition



Primal-dual step computation

- We can be brave and approach the full system

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

- ... or compute a normal step, then approach the perturbed system

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ -\nabla c_k^{\mathcal{E}\,T} v_k^x \\ -\nabla c_k^{\mathcal{I}\,T} v_k^x + d_k^s \end{bmatrix}$$

Primal-dual step computation

- We can be brave and approach the full system

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

- ... or compute a normal step, then approach the perturbed system

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu S_k^{-2} & 0 & -I \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e - \lambda_k^{\mathcal{I}} \\ -\nabla c_k^{\mathcal{E}\,T} v_k^x \\ -\nabla c_k^{\mathcal{I}\,T} v_k^x + d_k^s \end{bmatrix}$$

- How do we allow inexact solutions?
- How do we handle nonconvexity?

Scaling the system

- First, we scale the system

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu I & 0 & -S_k \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -S_k & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ S_k^{-1} d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix}$$

- The primal-dual matrix has nicer properties
- Along with slack reset, to maintain

$$s_k \geq \max\{0, c^{\mathcal{I}}(x_k)\},$$

allows easier infeasibility detection

Newton methods for nonlinear equations

- Newton's method

$$\mathcal{F}(x) = 0 \;\Rightarrow\; \nabla\mathcal{F}(x_k)d_k = -\mathcal{F}(x_k)$$

- Judge progress by the merit function

$$\phi(x) \triangleq \tfrac{1}{2}\|\mathcal{F}(x)\|^2$$

- Direction is one of descent since

$$\nabla\phi(x_k)^T d_k = \mathcal{F}(x_k)^T\nabla\mathcal{F}(x_k)d_k = -\|\mathcal{F}(x_k)\|^2 < 0$$

(Note the consistency between the step computation and merit function!)

Inexact Newton methods for nonlinear equations

- Compute

$$\nabla \mathcal{F}(x_k) d_k = -\mathcal{F}(x_k) + r_k$$

requiring (Dembo, Eisenstat, Steihaug (1982))

$$\|r_k\| \leq \kappa \|\mathcal{F}(x_k)\|, \quad \kappa \in (0,1)$$

- Progress judged by the merit function

$$\phi(x) \triangleq \tfrac{1}{2}\|\mathcal{F}(x)\|^2$$

- Again, note the consistency...

$$\nabla \phi(x_k)^T d_k = \mathcal{F}(x_k)^T \nabla \mathcal{F}(x_k) d_k = -\|\mathcal{F}(x_k)\|^2 + \mathcal{F}(x_k)^T r_k \leq (\kappa - 1)\|\mathcal{F}(x_k)\|^2 < 0$$

Merit function

- Simply minimizing

$$\varphi(x, s, \lambda^{\mathcal{E}}, \lambda^{\mathcal{I}}) = \tfrac{1}{2}\|\mathcal{F}(x, s, \lambda^{\mathcal{E}}, \lambda^{\mathcal{I}})\|^2$$

(where $\mathcal{F}$ is KKT error) is inappropriate for optimization

- We use the merit function

$$\phi(x, s; \pi) \triangleq f(x) - \mu \sum_{i=1}^{q} \ln s^i + \pi \left\| \begin{bmatrix} c^{\mathcal{E}}(x) \\ c^{\mathcal{I}}(x) - s \end{bmatrix} \right\|$$

where π is a penalty parameter

Model reductions

- Define the model of $\phi(x, s; \pi)$:

$$m(d^x, d^s; \pi) \triangleq f(x) + \nabla f(x)^T d^x - \mu \sum_{i=1}^{q} \ln s^i - \mu S^{-1} d^s$$
$$+ \pi \left(\left\| \begin{bmatrix} c^{\mathcal{E}}(x) + \nabla c^{\mathcal{E}}(x)^T d^x \\ c^{\mathcal{I}}(x) + \nabla c^{\mathcal{I}}(x)^T d^x - s - d^s \end{bmatrix} \right\| \right)$$

- d_k is acceptable if

$$\Delta m(d_k^x, d_k^s; \pi_k) \triangleq m(0, 0; \pi_k) - m(d_k^x, d_k^s; \pi_k) \gg 0$$

- This ensures descent (and more)

Termination test 1

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu I & 0 & -S_k \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -S_k & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ S_k^{-1} d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} + \begin{bmatrix} \rho^x \\ \rho^s \\ \rho^{\mathcal{E}} \\ \rho^{\mathcal{I}} \end{bmatrix}$$

The search direction is acceptable if

$$\left\| \begin{bmatrix} \rho^x \\ \rho^s \\ \rho^{\mathcal{E}} \\ \rho^{\mathcal{I}} \end{bmatrix} \right\| \leq \kappa \left\| \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} \right\| \quad \text{and} \quad \Delta m(d_k^x, d_k^s; \pi_k) \gg 0$$

Termination test 2

$$\begin{bmatrix} H_k & 0 & \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & \mu I & 0 & -S_k \\ \nabla c_k^{\mathcal{E}\,T} & 0 & 0 & 0 \\ \nabla c_k^{\mathcal{I}\,T} & -S_k & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ S_k^{-1} d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} + \begin{bmatrix} \rho^x \\ \rho^s \\ \rho^{\mathcal{E}} \\ \rho^{\mathcal{I}} \end{bmatrix}$$

The search direction is acceptable if

$$\left\| \begin{bmatrix} \rho^x \\ \rho^s \\ \rho^{\mathcal{E}} \\ \rho^{\mathcal{I}} \end{bmatrix} \right\| \leq \kappa \left\| \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} \right\| \quad \text{and} \quad \left\| \begin{bmatrix} \rho^{\mathcal{E}} \\ \rho^{\mathcal{I}} \end{bmatrix} \right\| \leq \epsilon \left\| \begin{bmatrix} c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} \right\|$$

Increasing the penalty parameter π then yields

$$\Delta m(d_k^x, d_k^s; \pi_k) \gg 0$$

Interior-point algorithm with inexact steps

(C., Schenk, and Wächter (2009))
for $k = 0, 1, 2, \ldots$

- Iteratively solve the primal-dual equations until termination test 1 or 2 is satisfied
- If only termination test 2 is satisfied, then increase π
- Backtrack from $\alpha_k \leftarrow 1$ to satisfy fraction-to-the-boundary and sufficient decrease conditions
- Update the iterate
- Reset the slacks

Convergence (inner iteration)

Assumption

The sequence $\{(x_k, s_k, \lambda_k^{\mathcal{E}}, \lambda_k^{\mathcal{I}})\}$ is contained in a convex set Ω over which f, $c^{\mathcal{E}}$, $c^{\mathcal{I}}$, and their first derivatives are bounded and Lipschitz continuous

Theorem

If all limit points of the sequence of constraint Jacobians have full row rank, then

$$\lim_{k\to\infty} \left\| \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} \right\| = 0.$$

Otherwise,

$$\lim_{k\to\infty} \left\| \begin{bmatrix} \nabla c_k^{\mathcal{E}} & \nabla c_k^{\mathcal{I}} \\ 0 & -S_k \end{bmatrix} \begin{bmatrix} c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} - s_k \end{bmatrix} \right\| = 0$$

and if $\{\pi_k\}$ is bounded, then

$$\lim_{k\to\infty} \left\| \begin{bmatrix} \nabla f_k + \nabla c_k^{\mathcal{E}} \lambda_k^{\mathcal{E}} + \nabla c_k^{\mathcal{I}} \lambda_k^{\mathcal{I}} \\ -\mu e - S_k \lambda_k^{\mathcal{I}} \end{bmatrix} \right\| = 0$$

Convergence (outer iteration)

Theorem

If the algorithm yields a sufficiently accurate solution to the barrier subproblem for each $\{\mu_j\} \to 0$ *and if the linear independence constraint qualification (LICQ) holds at a limit point* $\bar{x}$ *of* $\{x_j\}$*, then there exist Lagrange multipliers* $\bar{\lambda}$ *such that the first-order optimality conditions of the nonlinear program are satisfied*

Outline

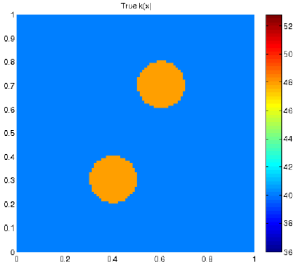
Implementation details

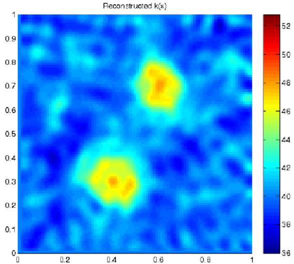
- Incorporated in IPOPT software package (Wächter)
  - inexact_algorithm yes
- Linear systems solved with PARDISO (Schenk)
  - SQMR (Freund (1994))
- Preconditioning in PARDISO
  - incomplete multilevel factorization with inverse-based pivoting
  - stabilized by symmetric-weighted matchings
- Optimality tolerance: 1e-8

CUTEr and COPS collections

- 745 problems written in AMPL
- 645 solved successfully
- 42 “real” failures
- Robustness between 87%-94%
- Original `IPOPT`: 93%

Helmholtz





N	n	p	q	# iter	CPU sec (per iter)
32	14724	13824	1800	37	807.823 (21.833)
64	56860	53016	7688	25	3741.42 (149.66)
128	227940	212064	31752	20	54581.8 (2729.1)

Boundary control

$$\begin{aligned} \min\ & \tfrac{1}{2}\int_\Omega (y(x) - y_t(x))^2 dx \\ \text{s.t.}\ & -\nabla \cdot (e^{y(x)} \cdot \nabla y(x)) = 20 \quad \text{in } \Omega \\ & y(x) = u(x) \quad \text{on } \partial\Omega \\ & 2.5 \leq u(x) \leq 3.5 \quad \text{on } \partial\Omega \end{aligned}$$

where

$$y_t(x) = 3 + 10x_1(x_1 - 1)x_2(x_2 - 1)\sin(2\pi x_3)$$

N	n	p	q	# iter	CPU sec (per iter)
16	4096	2744	2704	13	2.8144 (0.2165)
32	32768	27000	11536	13	103.65 (7.9731)
64	262144	238328	47632	14	5332.3 (380.88)

Original IPOPT with $N = 32$ requires 238 seconds per iteration

Hyperthermia treatment planning

$$\begin{aligned} \min\ & \frac{1}{2}\int_\Omega (y(x) - y_t(x))^2 dx \\ \text{s.t.}\ & -\Delta y(x) - 10(y(x) - 37) = u^* M(x) u \ \text{ in } \Omega \\ & 37.0 \le y(x) \le 37.5 \ \text{ on } \partial\Omega \\ & 42.0 \le y(x) \le 44.0 \ \text{ in } \Omega_0 \end{aligned}$$

where

$$u_j = a_j e^{i\phi_j}, \quad M_{jk}(x) =< E_j(x), E_k(x) >, \quad E_j = \sin(j x_1 x_2 x_3 \pi)$$

N	n	p	q	# iter	CPU sec (per iter)
16	4116	2744	2994	68	22.893 (0.3367)
32	32788	27000	13034	51	3055.9 (59.920)

Original IPOPT with $N = 32$ requires 408 seconds per iteration

Groundwater modeling

$$\begin{aligned}
\min\ & \tfrac{1}{2}\int_\Omega (y(x)-y_t(x))^2 dx + \tfrac{1}{2}\alpha\int_\Omega[\beta(u(x)-u_t(x))^2 + |\nabla(u(x)-u_t(x))|^2]dx \\
\text{s.t.}\ & -\nabla\cdot(e^{u(x)}\cdot\nabla y_i(x)) = q_i(x) \ \text{ in } \Omega, \quad i=1,\ldots,6 \\
& \nabla y_i(x)\cdot n = 0 \ \text{ on } \partial\Omega \\
& \int_\Omega y_i(x)dx = 0, \quad i=1,\ldots,6 \\
& -1 \le u(x) \le 2 \ \text{ in } \Omega
\end{aligned}$$

where

$$q_i = 100\sin(2\pi x_1)\sin(2\pi x_2)\sin(2\pi x_3)$$

N	n	p	q	# iter	CPU sec (per iter)
16	28672	24576	8192	18	206.416 (11.4676)
32	229376	196608	65536	20	1963.64 (98.1820)
64	1835008	1572864	524288	21	134418. (6400.85)

Original `IPOPT` with $N = 32$ requires approx. 20 hours for the first iteration

Outline

Summary

- We have a new framework for inexact Newton methods for optimization
- Convergence results are as good (and sometimes better) than exact methods
- Preliminary numerical results are encouraging

Future work

- Tune the method for specific applications
- Incorporate useful techniques such as filters, second-order corrections, specialized preconditioners
- Use (approximate) elimination techniques so that larger (e.g., time-dependent) problems can be solved
- Utilize mesh refinement/multi-grid methods

Thanks!!